Faster RooFitting

Automated Parallel Computation of Collaborative Statistical Models

Carsten D. Burgard

Patrick Bos, Stephan Hageböck, Vince Croft,
Wouter Verkerke, Inti Pelupessy, Jisk Attema

4th of November
CHEP 2019
Adelaide

RooFit

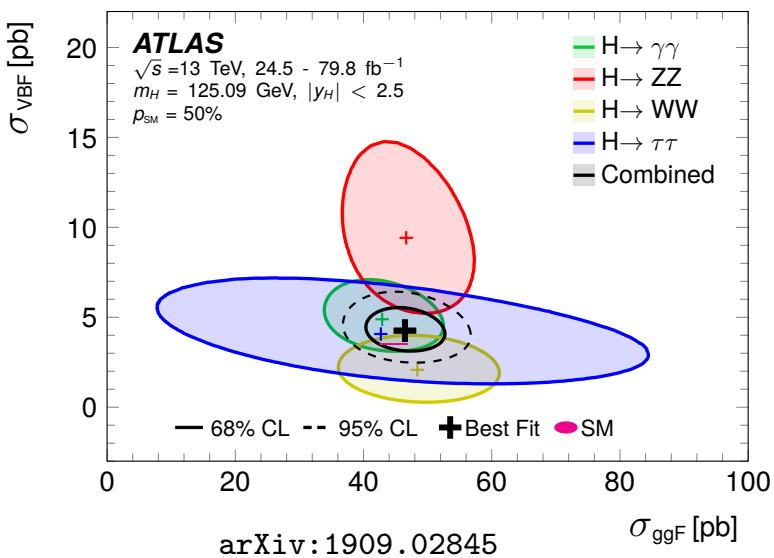
- `RooFit` is the statistical modeling & fitting package of `ROOT`
- used for statistical inference in many experiments
- extract parameters via Likelihood maximization
- separate model building & fitting from the technical implementation and **optimization**

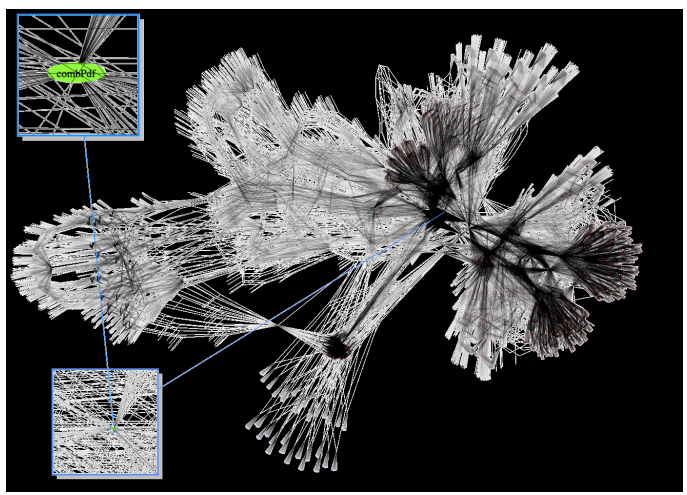
Parallelization

- `CPU` time dominated by repeated calculations of *-log L*
- `RooFit` internally parallelizes fitting
- user interface via `NumCPU(..)` argument

Collaborating & combining

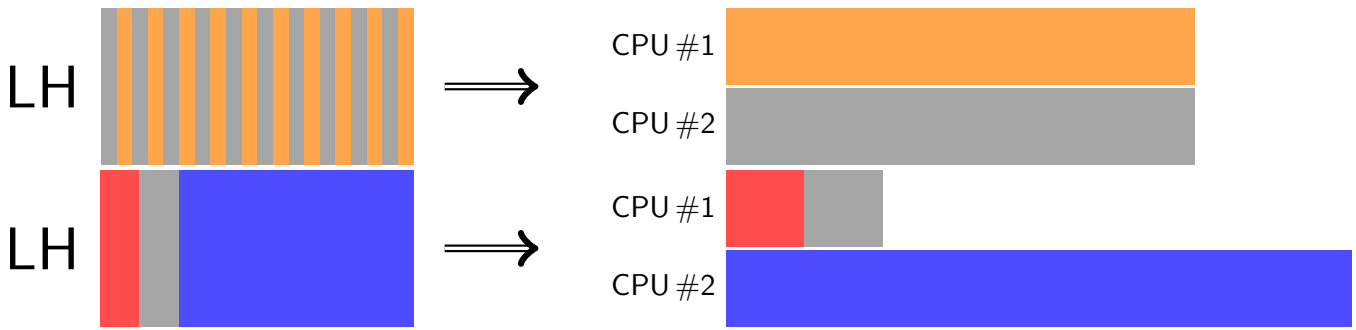
- central container object `RooWorkspace`
  - store models (p.d.f. modeling the data)
  - store datasets (binned or unbinned)
  - made persistent via `ROOT` streamer
- measurements can be easily combined by combining workspaces (both p.d.f and data)





RooFit: Existing Optimizations

- constant-term optimization/caching
- likelihood parallelization by event/bin



Faster RooFitting

- further parallelization (this project)
- vectorization of calculations (Stephan Hageböck)

Challenges

- Many components too small to efficiently distribute over many cores $\Rightarrow$ Relatively large overhead
- Other components very large (binned regions), not splitting efficiently over multiple workers

Solutions

1. Increase chunk sizes
2. Dynamic load balancing

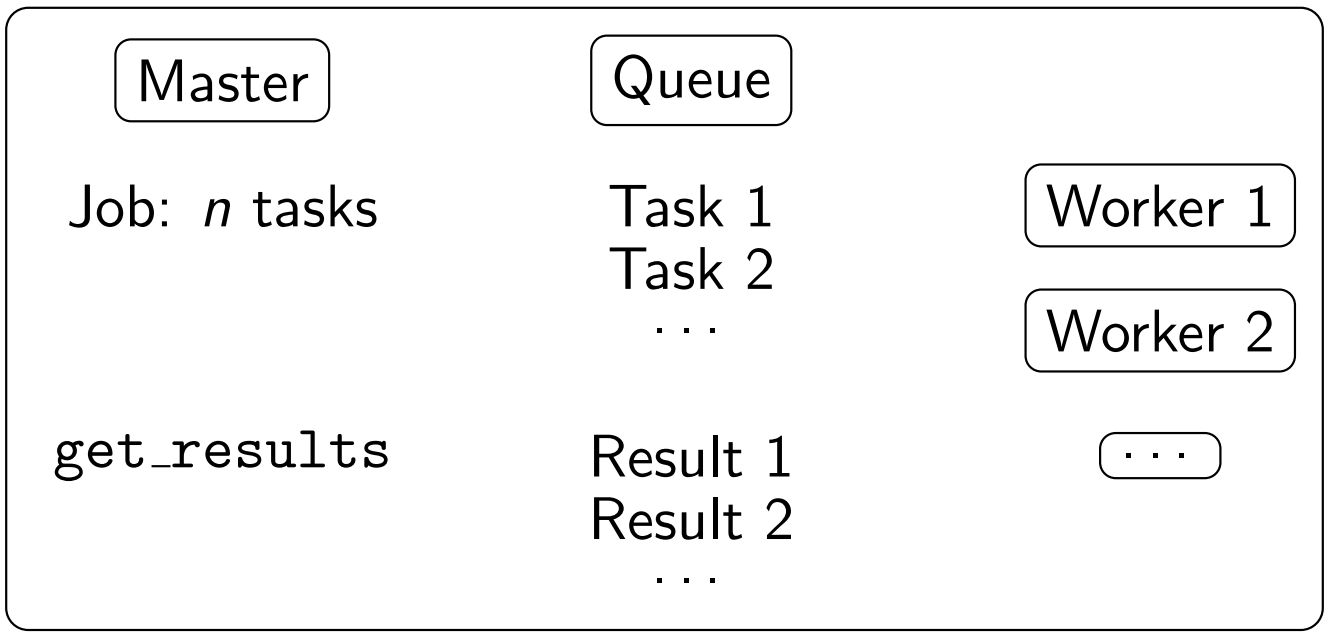
Solution 1: Partial derivatives

- Most CPU time spent in derivative calculations inside minimizer `MINUIT`
  - 1 Iteration $\sim$ Gradient + line-search
  - gradient for N parameters p: $\frac{df}{dp} \approx \frac{f(p-dp)-f(p)}{dp} \Rightarrow 2N$ calls of f
  - line-search: descend along gradient direction $\Rightarrow O(1)$ calls of f
- Focus on partial derivatives as *calculation chunk* instead of component likelihood
  - Required changes in `MINUIT`
  - made sure outputs stay exactly same

Solution 2: Dynamic Load Balancing

old custom `BidirMMapPipe` handles `fork`, `mmap`, pipes

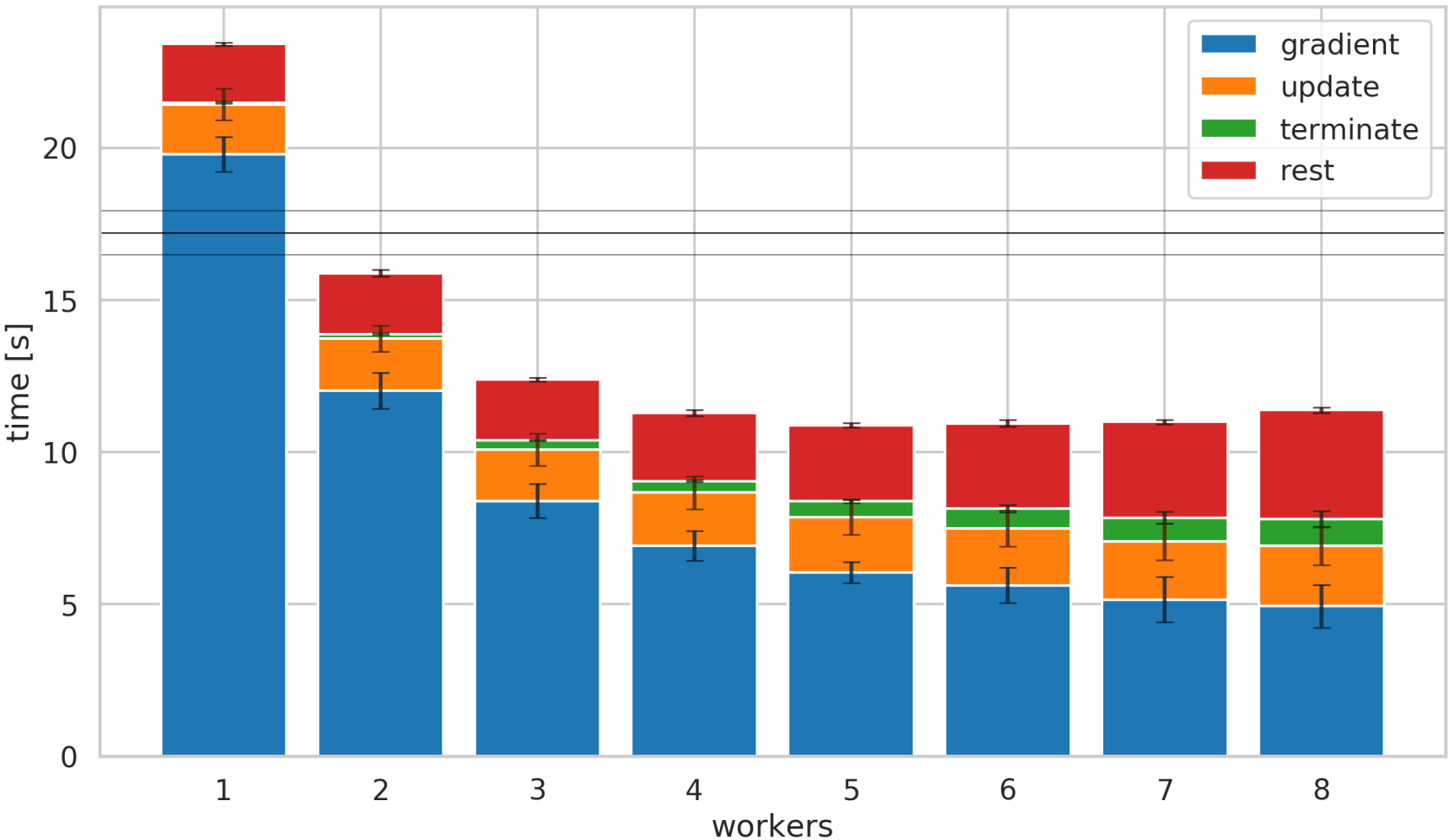
new ∅MQ for communication between forks



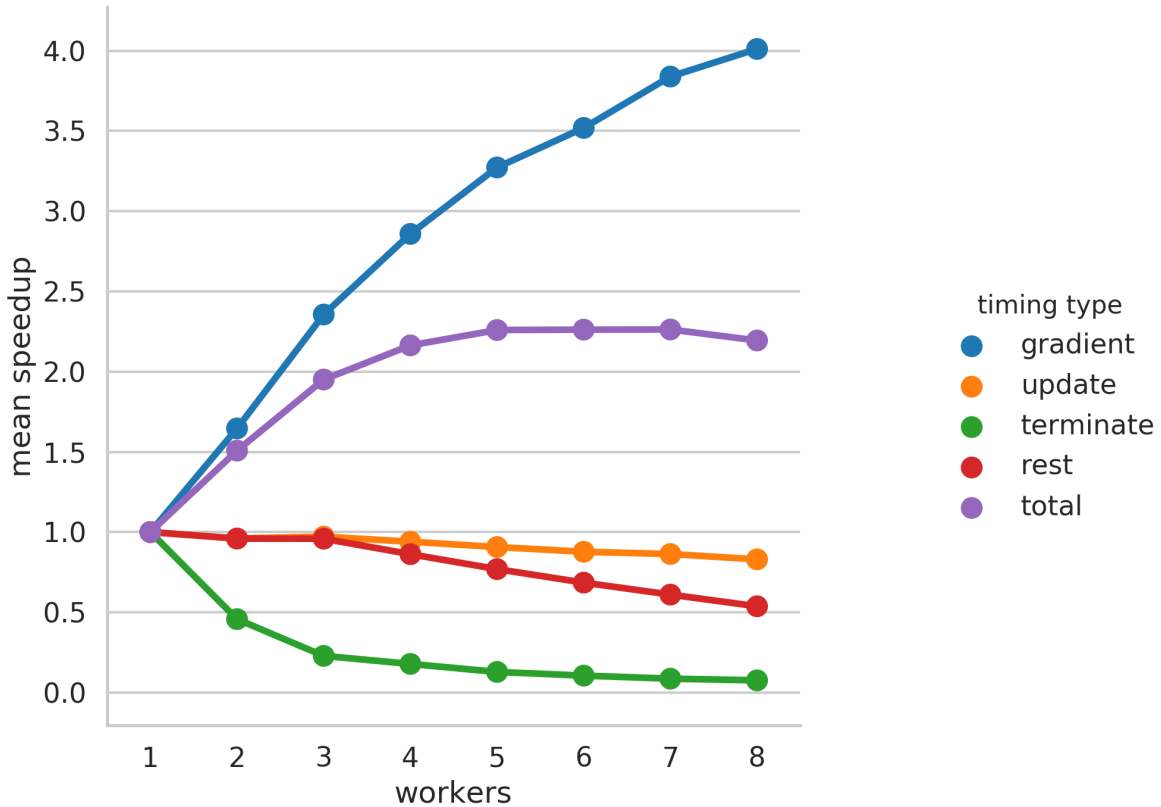
Benchmarks for MIGRAD fit / minimization

label	fast	big
model	ATLAS $H \to WW$ fit	ATLAS Higgs combination Moriond 2019
components	13795	126883
parameters	265	1487
approx. timing	20 s	10 min – few hours
comment	not main target audience, but used for fast benchmarking	depending on starting point

Small model

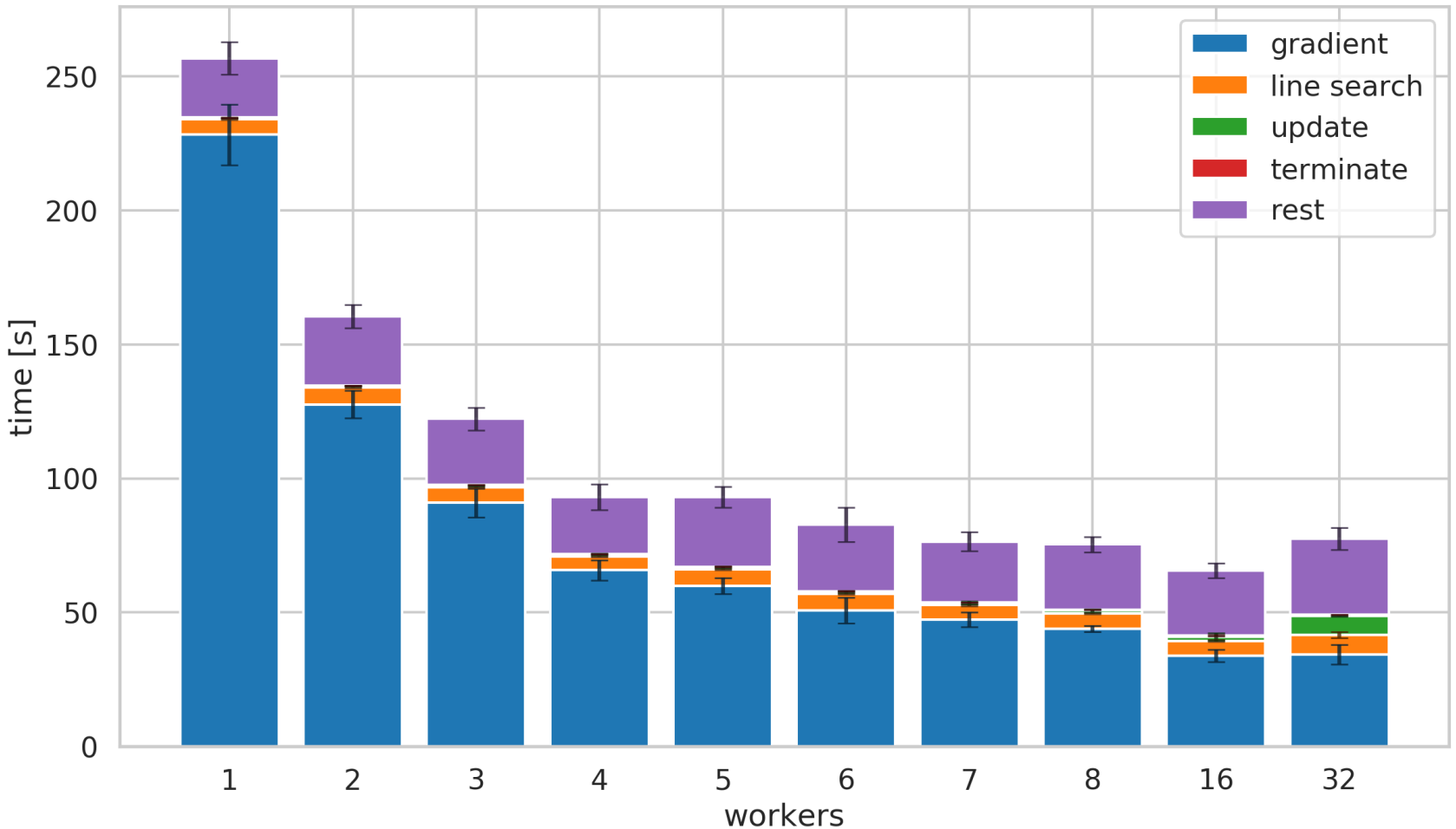

gradient
update
terminate
rest
time [s]
0
5
10
15
20
1
2
3
4
5
6
7
8
workers

Small model

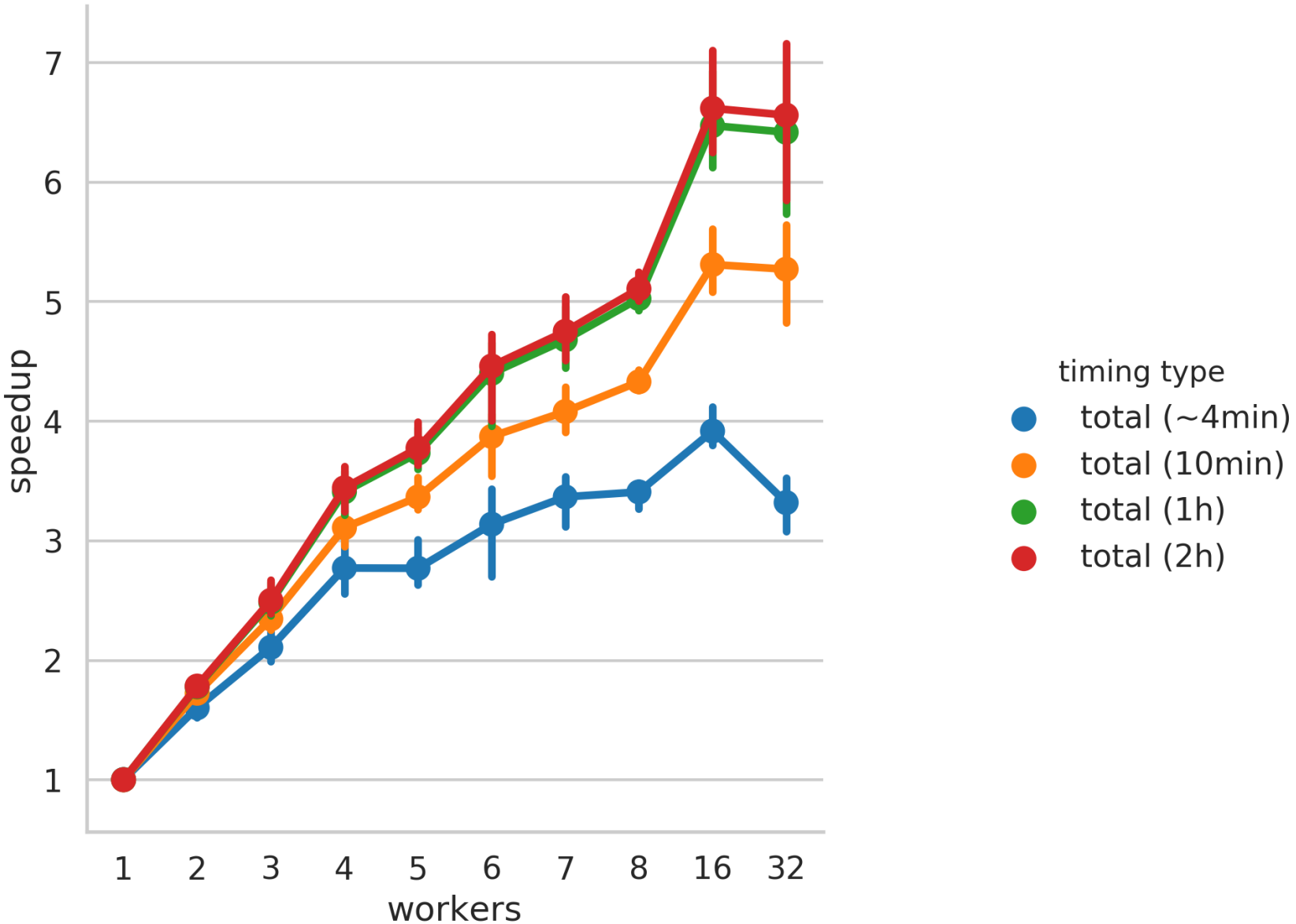


- update slow, but at least constant
  - fast model not "main target"
  - mainly want to speed up runs that take hours
  - this constant part becomes insignificant
- rest ignored for now, focusing on long runs

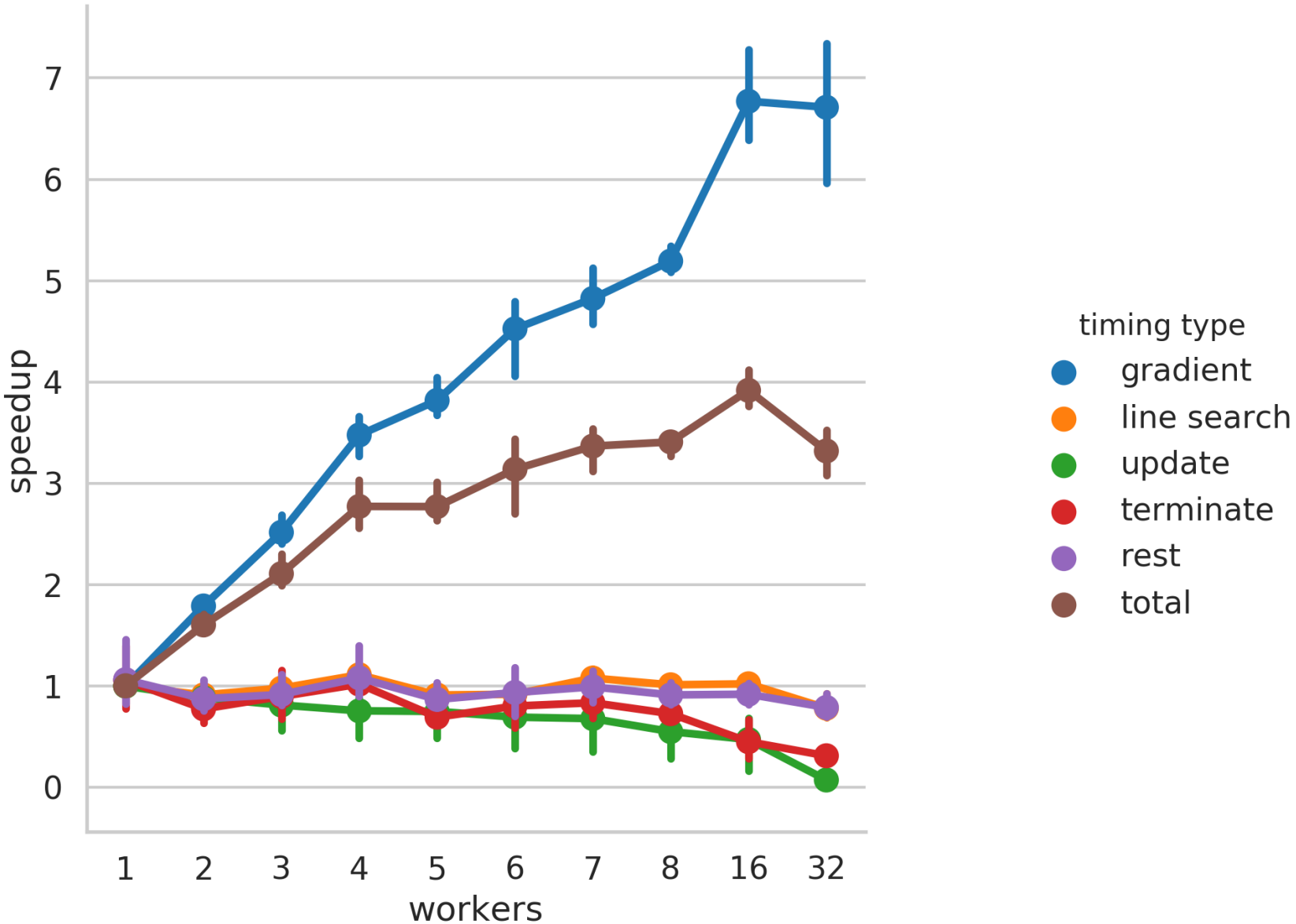
Big model


gradient
line search
update
terminate
rest
time [s]
0
50
100
150
200
250
1
2
3
4
5
6
7
8
16
32
workers

Big model

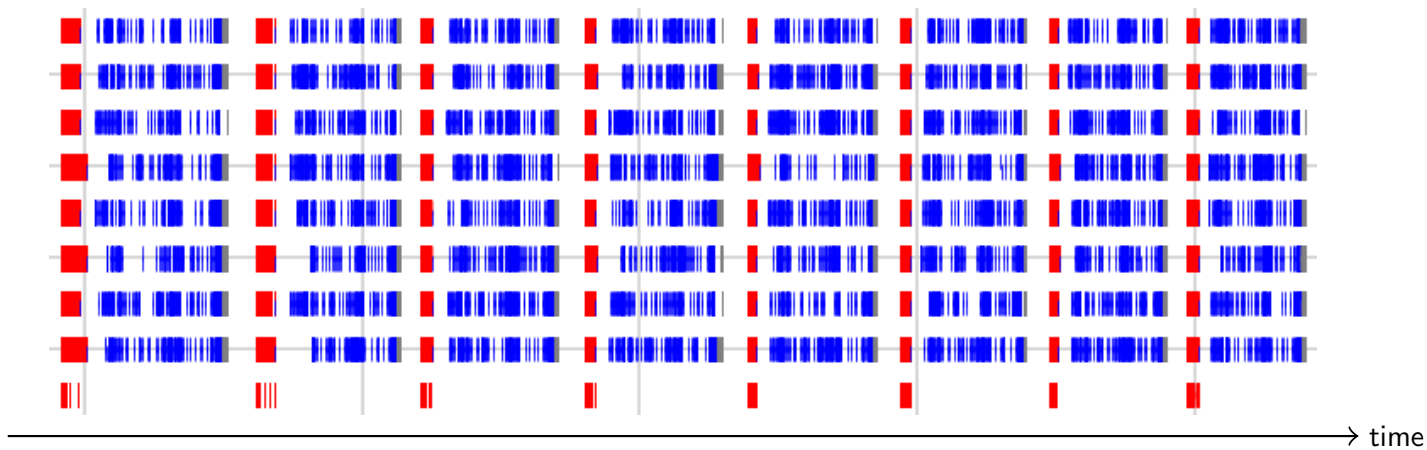

speedup
7
6
5
4
3
2
1
1
2
3
4
5
6
7
8
16
32
workers
timing type
total (~4min)
total (10min)
total (1h)
total (2h)

Big model


speedup
0
1
2
3
4
5
6
7
workers
1
2
3
4
5
6
7
8
16
32
timing type
gradient
line search
update
terminate
rest
total

Big model

- gradient not scaling well
  - mainly due to first partial derivative on each node taking long due to expensive precalculation



- big rest term caused by long synchronization step in serial part (master node) between roofit and minuit
- specifically the many constant terms in this model

Conclusions

- effective collaboration requires short wall times
  - minutes rather than hours
  - Parallelization can deliver this
- improved scaling of existing likelihood-level parallelization
- introduced new flexible framework
  - multi-level parallelization: likelihood, gradient
- gradient-level parallelization scales for large ($> 1\,\mathrm{h}$) fits
  - main goal achieved!

Related poster

Stephan Hageböck: *A faster, more accessible RooFit*

Outlook

- include new infrastructure into ROOT/RooFit
- for the adventurous: development version
  - **github.com/roofit-dev/root**
- investigate imperfect scaling observed in gradient calculation
- redesign core test statistic classes for future-proof interface
- allows to plug in any new types of calculation strategy
  - e.g. analytical derivatives
- `HESSE` can be parallelized in similar way
- With combination of all techniques expect speedups of factor 20 for fits > 1 hour.

What to expect in the future